Software Architecture Design Document

Mandatory Artifact for a web based centralized e-attendance System

Department of Information Technology, Government of Nepal

Version 1.1, 8-Aug-2021

Submitted by

XYZ CO

Document History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | Version | Author | Reviewed by | Approved by | Description |
| 2021-01-20 | 1.0 | XYZ | ABC |  |  |
| 2021-08-08 | 1.0 | XYZ | ABC |  |  |

Table of Contents

[1. Introduction 6](#_Toc81824548)

[1.1 Project Background 6](#_Toc81824549)

[1.2 Project Objectives 6](#_Toc81824550)

[1.3 Purpose of this Document 7](#_Toc81824551)

[1.4 Structure of this Document 7](#_Toc81824552)

[Introduction: 7](#_Toc81824553)

[Architecture: 7](#_Toc81824554)

[Architecture Components 7](#_Toc81824555)

[Infrastructure Architecture 8](#_Toc81824556)

[Security Design Process 8](#_Toc81824557)

[Security Infrastructure Architecture Design 8](#_Toc81824558)

[Infrastructure Security Architecture Design 8](#_Toc81824559)

[Justification of Architecture 8](#_Toc81824560)

[Technology Stack and Server Configuration 8](#_Toc81824561)

[Formatting 9](#_Toc81824562)

[2. Architecture 10](#_Toc81824563)

[2.1. Architecture Overview 10](#_Toc81824564)

[2.2. Physical Architecture Overview 11](#_Toc81824565)

[2.3. Logical Architecture Overview 14](#_Toc81824566)

[3. Architecture Components 18](#_Toc81824567)

[3.1. Data Model Overview 18](#_Toc81824568)

[3.1.1. Data Architecture Strategy 18](#_Toc81824569)

[3.1.2. Data Model 19](#_Toc81824570)

[3.1.3. Enhanced/Modified Data Stores 19](#_Toc81824571)

[3.1.4. Leveraged Data Stores 19](#_Toc81824572)

[3.1.5. Data Access 19](#_Toc81824573)

[3.1.6. Data Dictionary 19](#_Toc81824574)

[3.2. Detailed Data Model 19](#_Toc81824575)

[3.2.1 E-Attendance System 19](#_Toc81824576)

[3.2.2 Data Management 55](#_Toc81824577)

[3.2.3 Documentation and training impacts 56](#_Toc81824578)

[4. Infrastructure Architecture 57](#_Toc81824579)

[4.1 Infrastructure Architecture principles 57](#_Toc81824580)

[4.1.1 High availability 57](#_Toc81824581)

[4.1.2 Reliability 58](#_Toc81824582)

[4.1.3 Scalability 61](#_Toc81824583)

[4.1.4 Serviceability 61](#_Toc81824584)

[4.1.5 Manageability 62](#_Toc81824585)

[4.1.6 DR Planning 62](#_Toc81824586)

[4.2 Architectural Design 62](#_Toc81824587)

[4.2.1 Network Core 62](#_Toc81824588)

[4.2.2 Aggregation and Access Layer 62](#_Toc81824589)

[4.2.3 Service and Server Farms 62](#_Toc81824590)

[4.2.4 Load balanced services 63](#_Toc81824591)

[4.3 Server Infrastructure 63](#_Toc81824592)

[4.3.1 Deployment Architecture 63](#_Toc81824593)

[4.3.2 Storage Services 64](#_Toc81824594)

[4.3.3. Application Inventory 64](#_Toc81824595)

[4.3.4. Database Services 64](#_Toc81824596)

[5. Security Design Processes 65](#_Toc81824597)

[5.1 Activities 65](#_Toc81824598)

[5.1.1 Analysis Activities 65](#_Toc81824599)

[6. Security Infrastructure Architecture Design 67](#_Toc81824600)

[6.1 Security Policy 67](#_Toc81824601)

[6.2 Security Threats 68](#_Toc81824602)

[6.3 Network Security 69](#_Toc81824603)

[6.4 Software/Application Security 70](#_Toc81824604)

[6.5 Facilities Security 70](#_Toc81824605)

[7. Infrastructure Security Architecture Design 71](#_Toc81824606)

[8. Justification of architecture 72](#_Toc81824607)

[8.1. System architecture capabilities 72](#_Toc81824608)

[8.2. Network architecture capabilities 73](#_Toc81824609)

[9. Technology Stack and Server Configuration 74](#_Toc81824610)

[10. Standard and regulatory references 77](#_Toc81824611)

[11. Annexure 78](#_Toc81824612)

[11.1 Data Dictionary Template (Mandatory) 78](#_Toc81824613)

# 1. Introduction

This document has been created as a part of Artifacts Document. This Software Architecture Design Document (SDD), prescribed in GEA2.0 mandatory artifacts, documents the comprehensive overview of the architecture of the E-Attendance System.

## 1.1 Project Background

Most of the Government Organization here in Nepal have large number of employees working under various sections/departments/divisions/units etc. For all these employees, human resource department of organization needs to track their daily attendance which directly influence organization's productivity. Currently, in Department of Information Technology (DOIT) there is traditional way of logging the daily attendance manually in register sheet. This traditional method is so much time consuming and little bit hassle to manage. Thus, since Information Technology is now emerging like wave across the world contributing to make things easy to manage, DOIT is adopting the IT solution by Developing **"E-Attendance System".**

The E-Attendance System will help agency to see who is checked in and in what time. By doing this, we will be benefitted by keeping track of employee working hours. We will be ensured that we can manage employee's time effectively so that we can get most out of their time. We can maintain complete records of sick leaves and holidays, as well ensure that our employees are being paid correctly.

The E-Attendance System will include Employee Profile Management and Leave Management as Core Functional Module and will retrieve the Daily Attendance Data from Biometric Device using Web API.

## 1.2 Project Objectives

The main objective of this assignment is to develop, implement and support a centralized e-attendance system to manage employee’s attendance, leave, kaaj and profile for all government agencies which could also serve its purpose by controlling repetitive expenses on implementing same system at different government offices.

## 1.3 Purpose of this Document

This Software Architecture Design Document intends to provide a description of the design of a e-attendance system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to be built.

This document has been prepared using the scope of work outlined in the ToR, Functional Requirement Specification (FRS) and Software Requirement Specification (SRS) baseline and the information collected through elaborate workshops with all the relevant process owners and stakeholders.

## 1.4 Structure of this Document

The structure of this document has been broken to following for easier study.

### Introduction:

This section contains introductory part of the document such as project background, its objectives and purpose.

### Architecture:

This section contains application overview. Further, this section contains:

1. **Architecture Overview:** contains general description of the system from users point of view
2. **Physical Architecture Overview:** contains components diagrams, deployment diagrams, network diagrams, interface diagrams
3. **Logical Architecture Overview:** contains the top level software components and their interactions/relationships.

### Architecture Components

This sections contents the overview of the data model. Further this sections contains:

1. Data Model Overview
2. Data Architecture Strategy
3. Data Model
4. Enhanced/Modified Data Stores
5. Leveraged Data Stores
6. Data Access
7. Data Dictionary
8. Detailed Data Model
9. Data Management
10. Documentation and Training Impacts

### Infrastructure Architecture

This section contains information about infrastructure architecture related to this project. This section is divided into:

1. Infrastructure Architecture Principles
2. Architectural Design
3. Server Infrastructure

### Security Design Process

This section describe activities that are performed, the key types of work products produced, and resources that are used to support these activities.

### Security Infrastructure Architecture Design

This section contains infrastructure security architecture component details that comprise the Department's production and non-production technical infrastructure environment

### Infrastructure Security Architecture Design

This section contains information about information security architecture design.

### Justification of Architecture

This section contains rationale of the hardware / software architecture in terms of capabilities

### Technology Stack and Server Configuration

This section contains list of all the technology inclusive of hardware and software used

### Formatting

1. Style Declaration

**Font-family:** Georgia

**Font -size:** 12px

**Page Margin:** Top Left Bottom Right: 1 1 1 1

**List**: First: 1, 2,... ; Second: A,B,...; Third: I, II,...; Fourth: a, b, ...; Fifth: i, ii,....;

1. Headers

**Font Size :** H1: 28; H2: 18; H3: 16; H4: 14

**Font Style:** Bold, Italic
**Font Style:** Overline #c00000 (RGB 192,0,0), Text Color Black H1

**Color:** #c00000 (RGB 192,0,0) H2, H3, H4,

1. Document Body

**Paragraph**: 1.25 line spacing Before/After 12px.

**Table**: Border top/bottom #c00000 (RGB 192,0,0)

# 2. Architecture

## 2.1. Architecture Overview

The application e-Attendance System works in production environment in client side. The users of the application are:

1. Super Admin
2. Admin
3. Employee

The application aims to provide employee attendance management solution to the DOIT. It comprises of the below functions:

1. User Login
2. Interactive Dashboard
3. Leave Management (Request, Recommend, Approve)
4. Kaaj Management (Request, Approve)
5. Shift Management
6. Profile Management
7. Reports
8. System Configuration

The main interfaces with its inputs and outputs are given as below:

|  |  |  |
| --- | --- | --- |
| Interface Name | Inputs | Outputs |
| Login | Auth Credentials | Successful Login |
| Leave Apply | Data as required  | Successful Leave Apply |
| Leave Recommendation | Data as required | Successful Leave Recommendation |
| Leave Approve | Data as required | Successful Leave Approval |
| Kaaj Apply | Data as required | Successful Kaaj Apply |
| Kaaj Approve | Data as required | Successful Kaaj Approve |
| Shift Create | Data as required | Successful Shift Creation |
| Reports Generation | Filter data | Successful report generation |
| Configuration Add and Manage | Data as required | Successful addition of system configuration |

### 2.2. Physical Architecture Overview



Figure 1 Component Diagram



Figure 2 Deployment Diagram



Figure 3 Network Diagram



Figure 4 Activity Diagram

### 2.3. Logical Architecture Overview

The application "e-attendance system is going to be built in 3-tier architecture. The 3 tier approach for the application for application development depicts below information:



Figure 5 3-tier Architecture of e-attendance system

1. **Presentation Tier:** This tier generally is the front end layer and is the user interface of e-attendance system. This user interface is accessible through a web browser and it displays content and information related to the attendance application.
2. **Logic Tier:** This tier generally is the core business logic tier which processed the information collected in the presentation tier and against some information in data tier using core business rules. In logic tier, there is the list of classes which decide what information they need in order to solve assigned problems, request that information, manipulate that information as required, and return the ultimate results to the presentation layer for formatting.
3. **Data Tier:** This tier generally controls the servers where the information is stored, it runs a relational database management system on a database server and contains the application data storage logic. This tier usually keeps data independent from application servers or processing logic and improves scalability and performance.

The application "e-attendance system" will be based on 3 tier architecture and it will be generally built using MVC framework. MVC framework is an architectural pattern that separates an e-attendance application into three main logical components: Model, View and Controller.



Figure 6 General MVC Structure of e-attendance system

1. **Model**: The model encloses the clean application related data.
2. **View**: The view element is used for presenting the data of the model to the user. this element deals with how to link up with the model's data but does not provide any logic regarding what this data all about or how users can use these data.
3. **Controller**: The controller is generally in between the model and view element. it listens to all the incident and actions triggered in the view and performs an appropriate response back to the events.

**Folder Structure:**

E-Attendance system follows a standardized folder structure for better maintainability as given below:-

1. **Model:** All the models are located at project root/app/Models folder.
2. **View:** All of the e-attendance application views are located at *project\_root/resources/views* folder.
3. **Controller:** All of the e-attendance application controller are located at project*\_root/app/Http/Controller* folder.
4. **Repository:** All of the e-attendance application repositories are located at *project\_root/app/Repository* folder.



Figure 7 MVC Abstraction of e-attendance system

# 3. Architecture Components

## 3.1. Data Model Overview

### 3.1.1. Data Architecture Strategy

We have used below principles while designing the data model:

1. **Grain**: The most important data modeling concept is the grain of a relation. The grain of the relation defines what a single row represents in the relation. When designing a new relation , we:
2. Determined the grain of the relation
3. Named the relation such that the grain is clear
4. Ensured that all of the columns in the relation apply to the appropriate grain.
5. **Naming**: When it comes to naming data models, choose a naming scheme and stick with it.

Below are the relation naming standards:

1. Use schemas to name-space relations that are similar in terms of data source, business unit, or abstraction level.
2. Use the pluralized grain as the table name like

Below are the column naming standards:

1. IDs should get an \_id suffix, and primary keys should be called $OBJECT\_id
2. Timestamps should get a \_at suffix and dates should get a \_date suffix.
3. Booleans should begin with is\_ or has\_
4. **Materialization:** Materialization is most important tool for building a top-notch data model. By materialization, relation is created as a table or as a view. If we create relation as a table, we precompute any required calculations, which means that users will see faster query response times.
5. **Permissioning and governance:** In addition to determining the content of the data models and how the relations, as a data modelers we should be aware of the permissioning and governance requirements of the business.

### 3.1.2. Data Model

### 3.1.3. Enhanced/Modified Data Stores

*<Provide a list and a brief introduction of any modified data stores (databases) that are within the architecture-why the DB is being modified, etc.>*

***There is no such modified data stores within this application architecture***

### 3.1.4. Leveraged Data Stores

<Provide a list and a brief introduction of any data stores (databases) that are within the architecture – that exist and will not be modified.>

***There is no such leveraged data stores within this application architecture***

### 3.1.5. Data Access

Abstraction and Encapsulation used in programming for data access.

### 3.1.6. Data Dictionary

**Attached in Annex of this Document**

## 3.2. Detailed Data Model

### 3.2.1 E-Attendance System

#### 3.2.1.1 Entity- relationship diagram

#### 3.2.1.2 Tables

##### 3.2.1.2.1 Attendance logs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | Primary Key | YES | NO |  |
| device\_emp\_code |  | int(11) | Foreign Key | NO | NO |  |
| ip\_address |  | varchar(200) | Foreign Key | NO | NO |  |
| device\_id |  | int(11) | Foreign Key | NO | NO |  |
| log\_time |  | datetime |  | NO | NO |  |
| log\_types |  | tinyint(4) |  | NO | NO |  |
| created\_at |  | datetime |  | NO | NO |  |
| updated\_at |  | date |  | NO | NO |  |

##### 3.2.1.2.2 Attendances

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | Primary Key | YES | NO |  |
| emp\_id |  | int(11) | Foreign Key | NO | NO |  |
| shift\_id |  | int(11) | Foreign Key | NO | NO |  |
| device\_id |  | int(11) | Foreign Key | NO | NO |  |
| attendance\_id |  | date |  | NO | NO |  |
| in\_time |  | time |  | NO | NO |  |
| out\_time |  | time |  | NO | NO |  |
| create\_at |  | datetime |  | NO | NO |  |
| update\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.3 branch\_type

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| org\_id |  | int(11) |  | NO | NO |  |
| type\_name |  | varchar(255) |  | NO | NO |  |
| type\_short\_name |  | varchar(50) |  | NO | NO |  |
| active |  | Enum(‘Y’, ‘N’) |  | NO | NO |  |

##### 3.2.1.2.4 Branches

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| Id |  | int(11) | PK | YES | NO |  |
| Org\_id |  | int(11) | FK | YES | NO |  |
| Branch\_name |  | varchar(255) |  | YES | NO |  |
| Branch\_name\_np |  | varchar(255) |  | YES | NO |  |
| Created\_at |  | datetime |  | NO | NO |  |
| Updated\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.5 Countries

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| country\_name |  | varchar(255) |  | NO | NO |  |
| country\_name\_np |  | varchar(255) |  | NO | NO |  |
| country\_code |  | varchar(20) |  | NO | NO |  |
| country\_status |  | enum(‘Y’,’N’) |  | NO | NO |  |
| created\_At |  | datetime |  | NO | NO |  |
| updated\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.6 Designations

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| desg\_name |  | varchar(255) |  | NO | NO |  |
| desg\_name\_np |  | varchar(255) |  | NO | NO |  |
| desg\_order |  | nt(11) |  | NO | NO |  |
| desg\_status |  | enum(‘Y’,’N’) |  | NO | NO |  |
| created\_at |  | datetime |  | NO | NO |  |
| updated\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.7 Devices

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| org\_id |  | int(11) | FK | NO | NO |  |
| device\_name |  | varchar(255) |  | NO | NO |  |
| model\_name |  | varchar(255) |  | NO | NO |  |
| Ip\_address |  | varchar(20) |  | NO | NO |  |
| Port\_number |  | varchar(4) |  | NO | NO |  |
| Serial\_number |  | varchar(50) |  | NO | NO |  |
| Device\_status |  | enum(‘Y’,’N’) |  | NO | NO |  |
| Created\_at |  | datetime |  | NO | NO |  |
| Deleted\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.8 emplyee\_shifts

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| emp\_id |  | int(11) | FK | YES | NO |  |
| shift\_id |  | int(11) | FK | YES | NO |  |
| start\_date |  | date |  | YES | NO |  |
| end\_date |  | date |  | YES | NO |  |
| create\_at |  | datetime |  | NO | NO |  |
| update\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.9 employee\_type

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| Type\_name |  | varchar(255) |  | YES | NO |  |
| Type\_name\_np |  | varchar(255) |  | YES | NO |  |
| Type\_status |  | enum(‘Y’,’N’) |  | NO | NO |  |
| Created\_at |  | datetime |  | NO | NO |  |
| Updated\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.10 employees

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| org\_id |  | Int(11) | FK | YES | NO |  |
| org\_branch\_id |  | Int(11) | FK | YES | NO |  |
| loignid |  | Varchar(255) |  | YES | NO |  |
| User\_role\_id |  | Int(11) | FK | YES | NO |  |
| User\_lang\_id |  | Int(11) | FK | YES | NO |  |
| Device\_emp\_code |  | Int(11) |  | NO | NO |  |
| Emp\_sanket\_number |  | Int(11) |  | NO | NO |  |
| Desg\_id |  | Int(11) | FK | YES | NO |  |
| Fullname\_eng |  | Varchar(255) |  | YES | NO |  |
| fullname\_nep |  | Varchar(255) |  | YES | NO |  |
| gender |  | Enum(‘Y’,’N’) |  | YES | NO |  |
| birth\_place |  | Varchar(255) |  | NO | NO |  |
| mobile\_number |  | int(11) |  | NO | NO |  |
| email |  | Varchar(255) |  | NO | NO |  |
| marital\_status |  | Enum(‘Y’, ‘N’) |  | YES | NO |  |
| job\_appointment\_date |  | Date |  | NO | NO |  |
| lang\_id |  | Int(11) | FK | NO | NO |  |
| blood\_group |  | enum('A+','B+','AB+','O+','A-','B-','AB-','O-') |  | NO | NO |  |
| religion\_id |  | Int(11) | FK | YES | NO |  |
| dob |  | Date |  | NO | NO |  |
| complexion |  | Varchar(255) |  | NO | NO |  |
| is\_disabled |  | Enum(‘Y’, ‘N’) |  | YES | NO |  |
| about\_employee |  | Int(11) |  | NO | NO |  |
| photo |  | Varchar(255) |  | NO | NO |  |
| emp\_status |  | Enum(‘Y’, ‘N’) |  | NO | NO |  |
| emp\_type\_id |  | Int(11) | FK | NO | NO |  |
| created\_at |  | datetime |  | NO | NO |  |
| updated\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.11 kaaj\_applications

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| Id |  | Int(11) | PK | YES | NO |  |
| Emp\_id |  | Int(11) | FK | YES | YES |  |
| Emp\_org\_id |  | Int(11) | FK | NO | NO |  |
| Emp\_branch\_id |  | Int(11) | FK | NO | NO |  |
| Emp\_type\_id |  | Int(11) | FK  | NO | NO |  |
| Emp\_desg\_id |  | Int(11) | FK | NO | NO |  |
| Kaaj\_type\_id |  | Int(11) | FK | NO | NO |  |
| Year\_id |  | Int(11) | FK | NO | NO |  |
| Kaaj\_title |  | text |  | NO | NO |  |
| Kaaj\_details |  | text |  | NO | NO |  |
| Kaaj\_paid |  | enum('Y','N') |  | NO | NO |  |
| Is\_national |  | enum('Y','N') |  | NO | NO |  |
| Start\_date |  | date |  | NO | NO |  |
| End\_date |  | Date |  | NO | NO |  |
| Vehicle |  | Varchar(255) |  | NO | NO |  |
| Kaaj\_country\_id |  | int(11) | FK | NO | NO |  |
| Objective |  | Text |  | NO | NO |  |
| Venue |  | Text |  | NO | NO |  |
| Kaaj\_file |  | Varchar(255) |  | NO | NO |  |
| Kaaj\_status |  | Enum(‘Draft’, ‘Applied’, ‘Canceled’) |  | NO | NO |  |
| Approver\_id |  | Int(11) | FK | NO | NO |  |
| Approver\_status |  | Enum(‘Applied’, ‘Approved’, ‘Rejected’, ‘Canceled’) |  | NO | NO |  |
| Approver\_status\_date |  | Datetime |  | NO | NO |  |
| Applied\_date |  | Date |  | NO | NO |  |
| Created\_at |  | Datetime |  | NO | NO |  |
| Updated\_at |  | Datetime |  | NO | NO |  |

##### 3.2.1.2.12 kaaj\_types

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| Id |  | Int(11) | PK | YES | NO |  |
| Type\_name |  | Varchar(255) |  | NO | NO |  |
| status |  | Enum(‘Y’, ‘N’) |  | NO | NO |  |
| Created\_at |  | Datetime |  | NO | NO |  |
| Updated\_at |  | Datetime |  | NO | NO |  |

##### 3.2.1.2.13 languages

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| Id |  | Int(11) | PK | YES | NO |  |
| Lang\_name |  | Varchar(255) |  | NO | NO |  |
| Lang\_status |  | Enum(‘Y’, ‘N’) |  | NO | NO |  |
| Created\_At |  | Datetime |  | NO | NO |  |
| Updated\_at |  | Datetime |  | NO | NO |  |

##### 3.2.1.2.14 leave\_applications

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |   | int(11) | PK | YES | NO |   |
| emp\_id |   | int(11) | FK | YES | YES |   |
| emp\_type\_id |   | int(11) | FK | YES | NO |   |
| desg\_id |   | int(11) | FK | YES | NO |   |
| org\_id |   | int(11) | FK | YES | NO |   |
| org\_section\_id |   | int(11) |   | YES | NO |   |
| year\_id |   | int(11) | FK | YES | NO |   |
| leave\_apply\_date |   | date |   | YES | NO |   |
| leave\_type\_id |   | int(11) | FK | YES | NO |   |
| leave\_half\_day |   | enum('Y','N') |   | YES | NO |   |
| leave\_start\_date |   | date |   | NO | NO |   |
| leave\_end\_date |   | date |   | NO | NO |   |
| leave\_status |   | enum('Draft','Applied','Canceled') |   | YES | NO |   |
| leave\_file |   | varchar(255) |   | NO | NO |   |
| leave\_reason |   | text |   | NO | NO |   |
| recommender\_id |   | int(11) | FK | YES | NO |   |
| recommender\_status |   | enum('Applied','Approved','Rejected','Canceled') |   | NO | NO |   |
| recommender\_status\_date |   | date |   | NO | NO |   |
| recommender\_details |   | text |   | NO | NO |   |
| approver\_id |   | int(11) | FK | NO | NO |   |
| approver\_status |   | enum('Approved','Rejected') |   | NO | NO |   |
| approver\_status\_date |   | date |   | NO | NO |   |
| approver\_details |   | text |   | NO | NO |   |
| created\_at |   | date |   | NO | NO |   |
| updated\_at |   | date |   | NO | NO |   |

##### 3.2.1.2.15 leave\_balance

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| Id |  | Int(11) | PK | YES | NO |  |
| Emp\_id |  | Int(11) | FK | YES | YES |  |
| Year\_id |  | Int(11) | FK | YES | NO |  |
| Leave\_type\_id |  | Int(11) | Fk | YES | NO |  |
| Days\_number |  | Decimal(8,2) |  | YES | MO |  |

##### 3.2.1.2.16 leave\_titles

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | Int(11) | PK | YES | NO |  |
| title |  | varchar(255) |  | NO | NO |  |
| title\_short |  | varchar(50) |  | NO | NO |  |
| title\_np |  | varchar(255) |  | NO | NO |  |
| title\_short\_np |  | varchar(50) |  | NO | NO |  |
| status |  | enum('Y','N') |  | NO | NO |  |

##### 3.2.1.2.17 leave\_types

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| ord\_id |  | int(11) |  | NO | NO |  |
| leave\_title\_id |  | int(11) |  | NO | NO |  |
| male\_only |  | enum('Y','N') |  | NO | NO |  |
| fremale\_only |  | enum('Y','N') |  | NO | NO |  |
| contract\_eligible |  | enum('Y','N') |  | NO | NO |  |
| weekend\_countable |  | enum('Y','N') |  | NO | NO |  |
| public\_holiday\_countable |  | enum('Y','N') |  | NO | NO |  |
| saving\_leave |  | enum('Y','N') |  | NO  | NO |  |
| recurrance\_leave |  | enum('Y','N') |  | NO | NO |  |
| office\_head\_approval |  | enum('Y','N') |  | NO | NO |  |
| single\_leave |  | enum('Y','N') |  | NO | NO |  |
| deducate\_from\_other\_leave |  | enum('Y','N') |  | NO | NO |  |
| half\_day\_apply |  | enum('Y','N') |  | NO | NO |  |
| recommedner\_required |  | enum('Y','N') |  | NO | NO |  |
| satta\_bida |  | enum('Y','N') |  | NO | NO |  |
| active |  | enum('Y','N') |  | NO | NO |  |
| max\_saving |  | int(11) |  | NO | NO |  |
| total\_days |  | int(11) |  | NO | NO |  |
| number\_of\_times |  | int(11) |  | NO | NO |  |
| extra\_days |  | int(11) |  | NO | NO |  |
| apply\_docment\_within\_days |  | int(11) |  | NO | NO |  |
| approve\_within\_days |  | int(11) |  | NO | NO |  |
| created\_at |  | datetime |  | NO | NO |  |
| updated\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.18 leaves

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| org\_id |  | int(11) |  | NO | NO |  |
| leavel\_name |  | varchar(255) |  | NO | NO |  |
| level\_short\_name |  | varchar(255) |  | NO | NO |  |
| display\_order |  | enum('Y','N') |  | NO | NO |  |
| higer\_post |  | enum('Y','N') |  | NO | NO |  |
| middle\_post |  | enum('Y','N') |  | NO | NO |  |
| lower\_post |  | enum('Y','N') |  | NO | NO |  |
| active |  | enum('Y','N') |  | NO | NO |  |

##### 3.2.1.2.19 organization\_types

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| type\_name |  | varchar(255) |  | NO | NO |  |
| type\_name\_np |  | varchar(255) |  | NO | NO |  |
| type\_status |  | enum('Y','N') |  | NO | NO |  |
| created\_at |  | datetime |  | NO | NO |  |
| updated\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.20 organizations

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| org\_type\_id |  | int(11) | FK | YES | NO |  |
| org\_name |  | varchar(255) |  | NO | NO |  |
| org\_name\_np |  | varchar(255) |  | NO | NO |  |
| org\_code |  | varchar(20) |  | NO | NO |  |
| org\_code\_np |  | varchar(20) |  | NO | NO |  |
| main\_org\_id |  | int(11) |  | NO | NO |  |
| org\_fax |  | varchar(20) |  | NO | NO |  |
| org\_phone |  | varchar(50) |  | NO | NO |  |
| org\_pradesh\_id |  | int(11) |  | NO | NO |  |
| org\_address |  | varchar(255) |  | NO | NO |  |
| org\_address\_np |  | text |  | NO | NO |  |
| org\_remarks |  | text |  | NO | NO |  |
| org\_main\_office |  | enum('Y','N') |  | NO | NO |  |
| org\_status |  | enum('Y','N') |  | NO | NO |  |

##### 3.2.1.2.21 pradesh

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| pradesh\_name |  | varchar(255) |  | NO | NO |  |
| pradesh\_name\_np |  | varchar(255) |  | NO | NO |  |
| created\_at |  | datetime |  | NO | NO |  |
| updated\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.22 public\_holidays

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| org\_id |  | int(11) |  | NO | NO |  |
| holiday\_name |  | varchar(255) |  | NO | NO |  |
| holiday\_name\_short |  | varchar(50) |  | NO | NO |  |
| start\_date |  | date |  | NO | NO |  |
| end\_date |  | date |  | NO | NO |  |
| for\_male |  | enum('Y','N') |  | NO | NO |  |
| for\_female |  | enum('Y','N') |  | NO | NO |  |
| for\_only\_disable |  | enum('Y','N') |  | NO | NO |  |
| religion\_id |  | int(11) |  | NO | NO |  |
| active |  | int(11) |  | NO | NO |  |

##### 3.2.1.2.23 religions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| religion\_name |  | varchar(255) |  | NO | NO |  |
| religion\_status |  | enum('Y','N') |  | NO | NO |  |
| created\_at |  | datetime |  | NO | NO |  |
| updated\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.24 roles

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| role\_name |  | varchar(255) |  | YES | NO |  |
| role\_name\_np |  | varchar(255) |  | NO | NO |  |
| role\_status |  | enum('Y','N') |  | YES | NO |  |
| created\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.25 service\_types

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| org\_id |  | int(11) |  | NO | NO |  |
| service\_types |  | varchar(255) |  | NO | NO |  |
| service\_types\_short |  | varchar(255) |  | NO | NO |  |
| display\_order |  | int(11) |  | NO | NO |  |
| permanent\_available |  | enum('Y','N') |  | NO | NO |  |
| active |  | enum('Y','N') |  | NO | NO |  |

##### 3.2.1.2.26 shift\_days

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| shift\_id |  | int(11) | FK | NO | NO |  |
| day\_name |  | varchar(3) |  | NO | NO |  |
| start\_time |  | time |  | NO | NO |  |
| end\_time |  | time |  | NO | NO |  |
| is\_holiday |  | enum('Y','N') |  | NO | NO |  |
| created\_at |  | datetime |  | NO | NO |  |
| updated\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.27 shifts

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| org\_id |  | int(11) | FK | NO | NO |  |
| shift\_name |  | varchar(255) |  | NO | NO |  |
| shift\_status |  | enum('Y','N') |  | NO | NO |  |
| created\_at |  | datetime |  | NO | NO |  |

##### 3.2.1.2.28 years

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Mandatory | Unique | Source |
| id |  | int(11) | PK | YES | NO |  |
| year\_name |  | varchar(50) |  | YES | NO |  |
| year\_name\_np |  | varchar(50) |  | YES | NO |  |
| eng\_start\_date |  | date |  | YES | NO |  |
| eng\_nep\_date |  | datetime |  | YES | NO |  |
| running\_year |  | enum('Y','N') |  | YES | NO |  |
| year\_status |  | enum('Y','N') |  | NO | NO |  |
| created\_at |  | datetime |  | NO | NO |  |

#### 3.2.1.3 Indexes

##### 3.2.1.3.1 attendance\_logs

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | attendace\_logs | id | NO |
| device\_id | attendace\_logs | device\_id | NO |
| index\_device\_emp | attendace\_logs | device\_emp\_code | NO |
| index\_device\_emp | attendace\_logs | device\_id | NO |
| index\_ip\_address | attendace\_logs | ip\_address | NO |

##### 3.2.1.3.2 attendances

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | attendances | id | NO |
| shift\_id | attendances | shift\_id | NO |
| device\_id | attendances | device\_id | NO |
| index\_emp\_attendance\_date | attendances | emp\_id | NO |
| index\_emp\_attendance\_date | attendances | attendace\_date | NO |

##### 3.2.1.3.3 branch\_types

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | branch\_types | id | NO |

##### 3.2.1.3.4 branches

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | branches | id | NO |
| ord\_id | branches | ord\_id | NO |

##### 3.2.1.3.5 countries

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | countries | id | NO |

##### 3.2.1.3.6 designations

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | designations | id | NO |

##### 3.2.1.3.7 devices

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | devices | id | NO |
| org\_id | devices | org\_id | NO |

##### 3.2.1.3.8 employee\_shifts

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | employee\_shifts | id | NO |
| shift\_id | employee\_shifts | shift\_id | NO |
| emp\_id | employee\_shifts | emp\_id | NO |

##### 3.2.1.3.9 employee\_types

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | employee\_types | id | NO |

##### 3.2.1.3.10 employees

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | employees | id | NO |
| user\_lang\_id | employees | user\_lang\_id | NO |
| user\_role\_id | employees | user\_role\_id | NO |
| religion\_id | employees | religion\_id | NO |
| lang\_id | employees | lang\_id | NO |
| org\_branch\_id | employees | org\_branch\_id | NO |
| desg\_id | employees | desg\_id | NO |
| emp\_type\_id | employees | emp\_type\_id | NO |
| index\_emp\_org | employees | org\_id | NO |
| index\_emp\_org | employees | org\_branch\_id | NO |

##### 3.2.1.3.11 kaaj\_applications

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | kaaj\_applications | id | NO |
| kaaj\_type\_id | kaaj\_applications | kaaj\_type\_id | NO |
| kaaj\_country\_id | kaaj\_applications | kaaj\_country\_id | NO |
| emp\_org\_id | kaaj\_applications | emp\_org\_id | NO |
| emp\_branch\_id | kaaj\_applications | emp\_branch\_id | NO |
| emp\_type\_id | kaaj\_applications | emp\_type\_id | NO |
| emp\_desg\_id | kaaj\_applications | emp\_desg\_id | NO |
| year\_id | kaaj\_applications | year\_id | NO |
| index\_kaaj\_emp\_org | kaaj\_applications | emp\_id | NO |
| index\_kaaj\_emp\_org | kaaj\_applications | emp\_org\_id | NO |
| index\_year\_emp | kaaj\_applications | emp\_id | NO |
| index\_year\_emp | kaaj\_applications | year\_id | NO |
| index\_approver\_status | index\_approver\_status | approver\_id | NO |
| index\_approver\_status | index\_approver\_status | approver\_status | NO |

##### 3.2.1.3.12 kaaj\_types

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | kaaj\_types | id | NO |

##### 3.2.1.3.13 languages

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | languages | id | NO |

##### 3.2.1.3.14 leave\_applications

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | leave\_applications | id | NO |
| emp\_type\_id | leave\_applications | emp\_type\_id | NO |
| desg\_id | leave\_applications | desg\_id | NO |
| year\_id | leave\_applications | year\_id | NO |
| leave\_type\_id | leave\_applications | leave\_type\_id | NO |
| recommender\_id | leave\_applications | recommender\_id | NO |
| approver\_id | leave\_applications | approver\_id | NO |
| org\_id | leave\_applications | org\_id | NO |
| index\_leave\_emp\_org | leave\_applications | emp\_id | NO |
| index\_leave\_emp\_org | leave\_applications | org\_id | NO |
| index\_leave\_emp\_year | leave\_applications | emp\_id | NO |
| index\_leave\_emp\_year | leave\_applications | year\_id | NO |
| index\_emp\_year\_leave\_type | leave\_applications | emp\_id | NO |
| index\_emp\_year\_leave\_type | leave\_applications | year\_id | NO |
| index\_emp\_year\_leave\_type | leave\_applications | leave\_type\_id | NO |

##### 3.2.1.3.15 leave\_balances

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | eave\_balances | id | NO |
| year\_id | leave\_balances | year\_id | NO |
| leave\_type\_id | leave\_balances | leave\_type\_id | NO |
| index\_leave\_balance\_emp\_year\_type | leave\_balances | emp\_id | NO |
| index\_leave\_balance\_emp\_year\_type | leave\_balances | year\_id | NO |
| index\_leave\_balance\_emp\_year\_type | leave\_balances | leave\_type\_id | NO |
| index\_leave\_balance\_emp\_year | leave\_balances | emp\_id | NO |
| index\_leave\_balance\_emp\_year | leave\_balances | year\_id | NO |

##### 3.2.1.3.16 leave\_titles

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | leave\_titles | id | NO |

##### 3.2.1.3.17 leave\_types

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | leave\_types | id | NO |

##### 3.2.1.3.18 levels

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | leavels | id | NO |

##### 3.2.1.3.19 organization\_types

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | organization\_types | id | NO |

##### 3.2.1.3.20 organizations

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | Organizations | id | NO |
| org\_pradesh\_id | Organizations | org\_pradesh\_id | NO |
| index\_org\_pradesh | Organizations | org\_type\_id | NO |
| index\_org\_pradesh | Organizations | org\_pradesh\_id | NO |

##### 3.2.1.3.21 pradesh

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | pradesh | id | NO |

##### 3.2.1.3.22 public\_holidays

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | public\_holidays | Id | NO |

##### 3.2.1.3.23 religions

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | religions | id | NO |

##### 3.2.1.3.24 roles

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | roles | id | NO |

##### 3.2.1.3.25 service\_types

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | service\_types | Id | NO |

##### 3.2.1.3.26 shift\_days

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | shift\_days | int | NO |
| shift\_id | shift\_days | shift\_id | NO |

##### 3.2.1.3.27 shifts

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | shifts | id | NO |
| org\_id | shifts | org\_id | NO |

##### 3.2.1.3.28 years

|  |  |  |  |
| --- | --- | --- | --- |
| Index Name | Table | Column | Unique |
| PRIMARY | Years | Id | NO |

#### 3.2.1.4 Triggers

***There is no such triggers used in the application***

#### 3.2.1.5 Stored Procedures

***There is no such stored procedures written for this application.***

#### 3.2.1.6 Security

Below are the list of security threats that can occur:

1. Excessive privileges
2. Legitimate privilege abuse
3. Database injection attacks
4. Malware
5. Storage media exposure
6. Exploitation of vulnerable databases
7. Unmanaged sensitive data
8. The human factor

Below are mitigation factors that are included in security design for the proposed application:

1. Assessing for any database vulnerabilities, identifying compromised endpoints and classifying sensitive data
2. Managing user access rights and removing excessive privileges and dormant users
3. Monitoring all database access activity and usage patterns in real time to detect data leakage, unauthorized SQL and big data transactions, and protocol and system attacks
4. Blocking malicious web requests
5. Automating auditing with a database auditing and protection platform
6. Archiving external data and encrypting databases
7. Training employees on risk-mitigation techniques including how to recognize common cyber threats such as a spear-phishing attack, best practices around internet and email usage, and password management.

#### 3.2.1.7 Performance

We will ensure application database performance by implementing following measures:

1. **Monitoring availability and resource consumption :** This is the first step of proactive monitoring to check whether database is online at regular intervals. If nothing is offline, the next step will be to check resource consumption. In this case, resources are mainly infrastructure related like CPU, memory, disk, and network.
2. Measuring and Comparing Throughput: Throughput refers to the amount of work the database is doing under normal working conditions. Throughput measurement is part of regular monitoring.
3. Monitoring Expensive Queries
4. Tracking Database Changes
5. Monitoring Logs

#### 3.2.1.8 Capacity

1. The application will be scalable allowing additional modules to integrate.
2. The application will have capability to run on cross platform operating system
3. The application will be modular and loosely coupled
4. The application will be easily tested with automated tests
5. The application will have simple development and deployment
6. The application will have good architecture and pattern

#### 3.2.1.9 Data access

Abstraction and Encapsulation used in programming for data access.

#### 3.2.1.10 Error Handling

The debug option in config/app.php configuration file determines how much information about an error is actually displayed to the user. For local development, we set APP\_DEBUG environment variable to true. In production environment, this value will be made false since because if the value is set to true in production, we risk exposing sensitive configuration values to application end users.

The use of Exception Handler will be implemented where all the exceptions are handled by the App\Exceptions\Handler class. This class use two methods reports and render.

The report method is used to log exceptions or send them to an external service. By default the report method passes the exception to the base class where the exception is logged.

The render method is responsible for converting a given exception into an HTTP response that should be sent back to the browser. By default, the exception is passed to the base class which generates a response to us.

Below are the error message display standard that should be used in the e-attendance application.

|  |  |  |
| --- | --- | --- |
| Error Type | Description | Error Handling |
| Functionality Errors | Functionality error if something that we expect it to do is hard, awkward, confusing, or impossible. | Throwing an exception and handling them in a try catch block |
| Communications Errors | These errors occur in communication from software to end-user.  | Using users language Using visible and precise labelsIntegrating user manual for constructive help |
| Missing Command Errors | Occur when an expected command is missing. | Error handling and consistency definition for environment, platform |
| Syntax Error | Occur when there is missing or inappropriate syntax | Exceptional handlingGlobal correction |
| Error Handling Errors | Occur while the user is interacting with the software  | Exception Handling |
| Calculation Errors | Occur due to bad logic, Data Type Mismatch, Coding errors, Function Call issue etc. | Resolving Syntax errorStarting the debuggerIdentifying key variable and conditionsStepping to the suspicious codeLooking at the relevant variablesFixing it |
| Different functioning opposite to what a user has expected. |  | Passing through test cases and agile quadrant  |
| Missing functionality |  | Passing through test cases and agile quadrant  |
| Wrong functionality |  | Passing through test cases and agile quadrant  |
| Communication gap, while a user interacts with the system such as non-availability of on-screen instructions, etc. |  | Passing through test cases and agile quadrant  |
| Inappropriate text and also error message |  | Passing through test cases and agile quadrant  |
| Spelling, factual and likewise context error. |  | Passing through test cases and agile quadrant  |
| Uneven pattern of color |  | Passing through test cases and agile quadrant  |
| Incorrect and incomplete appearance of the statements. |  | Passing through test cases and agile quadrant  |
| Slow and poor responsive feature. |  | Passing through test cases and agile quadrant  |
| Error in the menu layout. |  | Passing through test cases and agile quadrant  |
| No revert function. |  | Passing through test cases and agile quadrant  |
| No back up facility |  | Passing through test cases and agile quadrant  |
| Incompatibility with the Operating system functions and also features. |  | Passing through test cases and agile quadrant  |
| Frequent time-out issues; and likewise, many such things |  | Passing through test cases and agile quadrant  |

#### 3.2.1.11 Installation and deployment strategy

**Pre-Installation Tasks:**

This is Web Based application. We have to be sure that the following thing is configured correctly in Linux based system

* Apache2
* MySql
* PHP and all its necessary module mentioned above.
* composer

Verifying everything is working fine in the server.

* Configure apache2 with document\_root**/var/www/<domain>**
* Install laravel and try to run *composer install* inside the laravel root directory if command fails than all the php module is not installed.
* After running *composer install* test the application from browser so that laravel will load its default index page without error

**Installation Procedure :**

Installation will be done on the following directory as standard. Developer will run the install process

* Application should be placed in /var/www/<domain>
* Run *composer install*
* Run following command for the file permission
	+ cd /var/www/<domain>
	+ sudofind . -type f -exec chmod 664 {} \;
	+ sudofind . -type d -exec chmod 775 {} \;
	+ *sudochgrp -R www-data storage bootstrap/cache*
	+ *sudochmod -R ug+rwx storage bootstrap/cache*
* Run the laravel migration or import the database for initial setup
* If the system is based on Redhat/Centos. System Admin with knowledge of related should be present

*Note*: For RedHat/Centos be sure to configure SeLinux in the beginning.

**Database:**

Configuration Modification:

Database Engine: MySql>=5.7

Users:

* Application will use the Privileged user that has full access to the Application Database
* If Database is installed in Application server than root user will use unix\_socket plugin to authenticate the database engine. Root user should only be used for database maintenance or configuration

Database creation /Update

* Database name should be start with application name
	+ Following command can be used to create database and user

*create database eattendance;*

*create user ‘eattendance’@’localhost’ identified by ‘password’;*

*grant all privileges on eattendance.\* to ‘eattendance’@’localhost’*

Data import/export

* For initial setup Application will run the migration and seed user for initial login.
* Database export can be done using the mysql tool **mysqldump**

#### 3.2.1.12 Data initialization

There are many instances where we need a populated database with test data in order to test the working of various operation. Populating test data manually is time consuming, but we use Laravel which provides a simple method of seeding test data in our database automatically.

We use database seeding in Laravel that helps to seed the database with test data using seed classes. Laravel initialize constant data that are inserted from seeder.

##### 3.2.1.12.1 Cleansing

Data cleaning is a crucial step in the analysis process where data is inspected to find anomalies, reduce repetitive data, and eliminate incorrect information. it does not involve deleting any existing information from the database; it focuses on enhancing the quality of data so that it can be used for analysis further.

We will include some of the best practices for data cleansing:

1. We will develop quality data plans to identify where maximum data quality errors occurs, to assess the root cause and design the plan accordingly
2. We will follow standard process of verifying the critical data before the creation of a database
3. We will identify any duplicates and validate the accuracy of the data to save time during analysis
4. We will track all the cleaning operations performed on the data

For above, we will use Laravel package to clean up models. The package named Laravel-model-cleaner helps to clean up unnecessary or old data. Models containing unneeded records will implement the provided 'GetsCleanedUp' -interface

|  |
| --- |
| use Spatie\ModelCleanup\GetsCleanedUp;use Illuminate\Database\Eloquent\Builder;use Carbon\Carbon;class LogItem extends Model implements GetsCleanedUp{ ... public static function cleanUp(Builder $query) : Builder { //delete up all records older than a year return $query->where('created\_at', '<', Carbon::now()->subYear()); }} |

##### 3.2.1.12.2 Conversion

##### *<Describe any data conversion that will be necessary to migrate the data – may include data types, mapping to completely new values, or adjusting other characteristics (i.e. length, etc.). Describe the tools to be used, processes, etc. If there is no conversion planned state there is none.>*

##### 3.2.1.12.3 Migration

*<Describe the data migration that will be necessary – if data will be manually entered or typed in prior to going into production describe that as well. Describe the tools to be used, processes, etc.>*

### 3.2.2 Data Management

We will recommend below mentioned data management activities:

1. **Backup**: This is the process of backing up the application data which is the valuable asset to the organization. We will develop automatic daily database backup. We will follow below steps for this:
2. Installing Laravel
3. Creating Command
4. Creating backup folder to store the DB
5. Scheduling Command

We can schedule the backup for daily, weekly, monthly basis.

1. **Log Maintenance:** We will maintain the proper logs for database. For this, we will create the database table that will store the change log data. Once the table has been created, any CREATE, ALTER or DROP commands that are run against a Table, Stored Procedure, Function or View within the database will be logged in the ChangeLog table going forward.
2. Support of disaster recovery : For the disaster recovery support, below steps can be implemented:
3. Backing up all data
4. Choosing the Right Backup Category : Full Backup, Incremental Backup, Differential Backup
5. Planning Effective Backup Strategy
6. Using necessary Data recovery Software
7. Documenting Critical Information
8. Testing and Rehearsing Disaster Recovery Plan

### 3.2.3 Documentation and training impacts

In the documentation and training for the technical users, we will include below contents:

1. Web Server
2. Programming
3. Database Backup
4. Framework
5. Coding
6. Security
7. Folder Structures
8. Administrative Controls
9. Language Files

# 4. Infrastructure Architecture

## 4.1 Infrastructure Architecture principles

Infrastructure architecture principles for e-attendance system is a structured and modern approach for supporting the "Department of Information Technology" and facilitating innovation within an enterprise. The architecture generally consists of activities responsible for ensuring the technical systems and infrastructure are designed to support business requirements. This may include support for business strategy in the ability to scale operations across different regions, support a business drive to support devices and to ensure data security.

### 4.1.1 High availability

The e-attendance system will have high availability characteristic which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.



Figure 8 High Availability Cluster

Here, applications can resubmit the transaction query to the secondary database pool if the primary does not respond.

High availability cluster shown above uses multiple systems that are already installed, configured and plugged in so that if a failure causes one of the systems to fails, another can be seamlessly leveraged to maintain the availability of application service being provided.

### 4.1.2 Reliability

Proposed application " e-attendance application" ensures the application reliability by checking the reliability in terms of reliability metrics.

Figure 9 Reliability Measurement Component

The application will be checked for its reliability based on the standard indicators used in the reliability metrics as below:

|  |  |  |
| --- | --- | --- |
| Metric | Purpose | Metric Formula |
| System Average Interruption Duration Index (SAIDI) | Indicator of sustained application breakdowns experienced by users | Total users minutes of application breakdown/Total number of users |
| System Average Interruption Frequency index (SAIFI) | Indicator of how many application breakdowns are experienced by users | Total number of application breakdowns/Total number of users |
| Customer Average Interruption Duration Index (CAIDI) | Indicator of the length of application breakdowns experienced by users | Total minutes of sustained user from app breakdown/total number of application breakdown |
| Momentary Average Interruption Frequency Index (MAIFI) | Indicator of momentary application breakdowns experienced by users | Total number of momentary application breakdown per year/total number of users |
| Application Quality | Indicator of Security Implementation and Requirement mapping which can cause deviation in development | SRS, FRS and Quality Report Check |



Figure 10 Reliability Engineering

### 4.1.3 Scalability

The proposed "e-attendance system" ensures scalability as the application development will implement below mentioned design principles:

1. Using correct tool with scalability in mind
2. Using multiple levels of caching in order to minimize the risk of Cache Miss
3. Using multiple databases to maintain data consistency and integrity
4. Avoiding using local disks and use object storage API to make storage manageable
5. Making web applications stateless unless there is a very good reason to have state or store sessions
6. Using asynchronous communication wherever possible to avoid wastage of application resources
7. Using queues to make tasks atomic so that they can be retrieved easily when fails
8. Maintaining ability to roll back code in case of failed deploys
9. Carrying our load or performance testing of application every single day
10. Rooting out any single point of failure (SPOF), when failure strikes application
11. Automating everything to run continuously

### 4.1.4 Serviceability

Serviceability is the measure of and the set of the features that support the ease and speed of which corrective maintenance and preventive maintenance can be conducted in application.

The proposed application ensure serviceability through below services:

1. **Corrective Maintenance (CM) :** This includes all the actions taken to repair failed system and get it back into an operating or available state. Since failure can be unexpected or expected but it is usually a unplanned outage, we use Mean Time To Repair (MTTR) which is the measure used to quantify the time required to perform CM and is also used in determining a system's availability.
2. **Preventive Maintenance (PM):** This includes all the actions taken to replace, service, upgrade or patch a system to retain its operational or available state and prevent system failures.

### 4.1.5 Manageability

For application manageability, we propose the flexible configuration architecture within the web application framework that tends to provide application manageability. This helps efficiently and easily manage the application that can be monitored and maintained to keep the system performing, secure and running smoothly.

### 4.1.6 DR Planning

For disaster recovery, we propose virtualized disaster recovery plan. Virtualization provides opportunities to implement disaster recovery in a more efficient and simpler way. A virtualized environment can spin up new virtual machine instances within the minutes and provide application recovery through high availability. Testing can also be easier to achieve.

## 4.2 Architectural Design

### 4.2.1 Network Core

***Since this application will be deployed in the GIDC server and all the server and network monitoring and management task will be done by the GIDC officials.***

### 4.2.2 Aggregation and Access Layer

***Since this application will be deployed in the GIDC server and all the server and network monitoring and management task will be done by the GIDC officials.***

### 4.2.3 Service and Server Farms

***Since this application will be deployed in the GIDC server and all the server and network monitoring and management task will be done by the GIDC officials.***

### 4.2.4 Load balanced services

***Since this application will be deployed in the GIDC server and all the server and network monitoring and management task will be done by the GIDC officials.***

## 4.3 Server Infrastructure

### 4.3.1 Deployment Architecture



Below are the pipeline of deployment architecture:

1. A deployment pipeline created, which is triggered when a commit is made to the referenced branch of the Github repository used in the source stage
2. In build/test stage, testing will be done
3. In deploy stage, the source branch of the Github repository is merged with the release branch

**Architecture Summary:**

1. Standard File Location Path

var/www/<domain>

1. Apache 2

/etc/apache 2/

vhost configuration:

/etc/apache2/sites-available/vhostshareconf/etc/apache2/conf-available

Apache 2 module conf:

/etc/apache2/mods-available

1. Mysql

/etc/mysql/

Mysql daemon conf file :

/etc/mysql/mysql.conf.d/mysld.conf (all domain related and mysql global variable will be set in this file)

1. PHP

In case of PHP-FPM

configuration file

/etc/php/<version>/fpm/

php-fpm daemon related conf path

/etc/php/<version>/pool.d/www.conf

### 4.3.2 Storage Services

***Since storage service is cloud based service, this section will not be applicable for us.***

### 4.3.3. Application Inventory

|  |  |  |
| --- | --- | --- |
| Version | Date | Modifications |
| 1.0 | 15/05/2021 | Basic Software Inventory-original file |
| 2 | 01/05/2021 | Print button added in the list page |
| 2.1 | 23/05/2021 | Leave Request form modified  |

### 4.3.4. Database Services

Our database service includes application database being protected and monitored by establishing backup and recovery procedures, providing a secure database environment and monitoring database performance.

# 5. Security Design Processes

This section describes the approach to application development that seeks to make application as free of vulnerabilities and impervious to attack as possible through such measures as continuous testing, authentication safeguards and adherence to best programming practices.

## 5.1 Activities

The following sections describe activities that are performed and resources that can be used to support these activities. [[1](#bra13)]

### 5.1.1 Analysis Activities

1. **Identifying Assets:** Most application are built on a frameworks such as laravel. These frameworks should be part of the asset's risk profile. The e-attendance application utilize database back ends, databases will also be included in assets identification. Other factors such as vectors (is it exposed to the internet or just internally accessible?) and existing protections will also be considered. We recommend implementation of well configured Web Application Firewall which can significantly affect the risk profile
2. **Identifying Threats:** This is the next crucial steps which includes the use of data discovered during asset profile creation to determine what threats may exist for the given system. Based on the data from the profiles, we can identify certain specific threats to the application. The identification of these threats usually involves the way in which an attacker would likely interact with a application. In our application, employee data would be at high risk of Web-and data-base-based attacks. Access to that employee data would be a threat; vulnerability (SQL injection) combined with a threat source (hacker) would be a realization of that threat.
3. **Identifying Vulnerabilities :** This is probably one of the most challenging aspects of risk assessment process. We will use automated tools such as network scanners, application scanners or remote agents to programmatically crawl over the application footprints and indentify vulnerabilities. We will also use human interaction and manual review to indentify other possible vulnerabilities.
4. Develop Metrics : We will use below table to determine the risk .

|  |  |  |  |
| --- | --- | --- | --- |
| Impact Severity | Impact Low-0.1 | Impact medium-0.5 | Impact High-1.0 |
| Likelihood of risk realization |
| Likelihood High-1.0 | 0.1 | 0.5 | 1.0 |
| Likelihood Medium-0.5 | 0.05 | 0.25 | 0.5 |
| Likelihood Low-0.1 | 0.01 | 0.05 | 0.1 |

This metric has immense value to security, audit and risk teams because it allows for a consistent scale that gauges the mathematical risk of a given vulnerability as it is applied to the asset.

1. **Considering Historical Breach Data:** It is extremely important to take the lessons regarding cost and impact from past breaches and apply that to our risk formulas.
2. **Calculating Cost:** Using the impact severity matrix, we can establish a certain risk and apply that to the cost factors. For eg; lets take SQL Injection, since we know that SQL injection in most cases is levied against Web interfaces with database back ends, we can apply this risk formula to a system identified as a web application and a database.
3. **Performing Fluid Risk to Asset Tracking :** Security risk assessment must remain fluid, continuing to take into account the ever changing threat landscape. For this, we use a method of tracking threats called fluid -risk-to-asset tracking.

# 6. Security Infrastructure Architecture Design

## 6.1 Security Policy

Below are the design considerations and policy enhancements recommendations within the department environment:

1. All the information security responsibilities need to be defined and allocated. It can be general ( protecting information) and/or specific (eg: the responsibility for granting particular permission)
2. Consideration should be given to the ownership of information assets or groups of assets when identifying responsibilities
3. Clarifying specific information security responsibilities within existing job roles
4. Conflicting duties and areas of responsibility must be segregated in order to reduce the opportunities for unauthorized or unintentional modification or misuse of any of the organization assets
5. Appropriate contacts with special interest groups or other specialist security forums and professional associations must also be maintained
6. Information security needs to be addressed in the project management, regardless of the type of project. We recommend the use of template frameworks for projects that include a simple repeatable checklist to show that information security is being considered.
7. The access control policy must be established, documented and reviewed regularly taking into account the requirements of the business for the assets in scope.
8. Clarify who needs to access, know, who needs to use the information-supported by documented procedures and responsibilities.
9. Access control rules should be supported by formal procedures and defined responsibilities.
10. A formal user registration and deregistration process needs to be implemented. A good process for user ID management includes being able to associate individual IDs to real people, and limit shared access ISs, which should be approved and recorded where done.
11. A process must be implemented to assign or revoke access rights for all user types. provisioning and revoking process should include: Authorization from the owner of the application for the use of information service. Verifying that the access granted is relevant to the role being done, and protecting against provisioning being done before authorization is complete.
12. Access to information and application system functions must be tied into the access control policy. Key considerations should include:
13. Role-based access control
14. Levels of access
15. Design of "menu" within the applications
16. Read, write, delete and execute permissions
17. Limiting output of information
18. Physical and/or logical access controls to sensitive applications, data and systems.
19. Access to systems and applications must be controlled by a secure log-on procedure to prove the identity of the user.
20. Access to program source code must be restricted. Controls should include consideration for:
21. As few people as possible having access
22. Keeping source code off operational systems (only compiled codes)
23. Access to source code being as restricted as possible (deny-by-default)
24. Access to source code being logged and the logs periodically reviewed.
25. Strong and strict change control procedures
26. Frequent audits and reviews.

## 6.2 Security Threats

Below are the design considerations that will address indentified threats or vulnerabilities:

1. Securing the weakest link within the application.
2. Access should be denied by default and only permitted when explicit permission exists
3. Every access to every objects must be checked for authority
4. The security of mechanism should not be depend on the secrecy of its design or implementation
5. Access should be granted based on more than one piece of information.
6. Every process and user should be given the least set of privileged that it needs in order to complete its task
7. The protection mechanism should be shared as little as possible among users
8. The protection mechanism should be easy to use.

## 6.3 Network Security

Below are the list of network security design considerations for this application that support network security improvements:

1. ISP Router: The Internet Service Provider (ISP) will have a router under its control that can implement security functions as DOIT policy requires. The ISP must provide department with below functions:
2. Ingress/egress filtering
3. Unicast RPF Unicast Reverse Path Forwarding
4. DDOS best practices.
5. Branch Versus Head-End Design Considerations: Each of the location of organization will have a network edge at the point of connection between them. Thus, the location is a head end ( central site) with full services required because they are generally more complex designs.
6. One of the principal considerations in remote site security deployments is management.
7. Remote site must be restricted from accessing certain central resources. This can be done for remote site first and then again at the central site router, which should be under closer watch.
8. Connection to the central site should be provided by a VPN connection over the public internet
9. Services like email that need to provide to outside users should be provide locally to reduce the strain on the central system.
10. Limit remote access to the network since the system using these remote connections were not as tightly controlled as the ones at the central site
11. Build a strong defensive perimeter around the boundary of the wired network, and host the mission critical computers and servers within the wired network. Standard techniques such as using firewalls and intrusion detection tools can be applied.
12. Deploy the wireless infrastructure outside the perimeter of the wired network. This will prevent inherent weakness in WLAN security from creating vulnerabilities in the defensive perimeter of the wired network
13. Protect the RF links used to carry information from between access points and the computer. The link has to be secured to protect the information and data that is transmitted over the airwaves.

## 6.4 Software/Application Security

The project built up in laravel development framework which uses below security packages:

1. Laravel Security Component: This provides security for the roles/objects and integrates Symphony security core in laravel. It uses voters to check role based privileges to different roles, so could validate its security
2. Laravel Security: Laravel security removes XSS vulnerabilities in the codebase.
3. laravel -ACL: laravel ACL provides role based secured permissions to the Laravel authentication process. This package helps protecting routes and CRUD controller methods in the applications.

Laravel Security Features are listed below:

1. Laravel Authentication System
2. Reduce Laravel Vulnerabilities from CSRF (Cross Site Request Forgery)
3. Protection against XSS (Cross Site Scripting )
4. SQL Injection
5. Improve Laravel Application Security
6. Laravel Security Packages

## 6.5 Facilities Security

Design considerations that support physical security improvements are as below:

1. Secure areas need to be protected by the appropriate entry controls to ensure only authorized personnel are allowed access
2. Identifying potential risks (both naturally -occurring and man-made), implementing physical protection against natural disasters, malicious attacks or accidents

***Since the attendance system will be hosted in the GIDC, all the facilities security will be maintained by them.***

# 7. Infrastructure Security Architecture Design

1. Information stored, created, or manipulated by the system may or may not be subject to an official classification that defines its sensitivity and the obligations to which the system and its owners are subject
2. Consideration must be made for different legislative burden that may hold jurisdiction over the system and the data stored
3. All the assets of value are kept and maintained by the DOIT. The specific person from DOIT charged with this responsibility must be indentified
4. In the event of system failure or loss of functionality, some value is lost to stakeholders. The cost of this opportunity loss should be quantified, if possible, and documented
5. Security is enhanced when security related changed can be implemented inexpensively and are, hence not sidelined.
6. Security is also enhanced when changes require no changes to code, changes to code introduce bugs and bugs introduce security vulnerabilities.

# 8. Justification of architecture

## 8.1. System architecture capabilities

The e-attendance application will have all the features listed in the business requirements along with the structured architecture. Justification of the system architecture capabilities is integrated in Programming Language Choice, Framework Choice, Application Type Choice and the Architecture Patterns.

1. Programming Language Choice: Application is built in PHP which is server side scripting language. PHP helps in enhancing the application capabilities as mentioned below:
2. Extremely flexible
3. It has easy integration and compatibility
4. It delivers efficient performance
5. Cost efficient
6. Giver web developer more control
7. It has better loading speed
8. It has more options for database connectivity
9. Framework Choice: Application is built in Laravel Framework which enhance application capabilities as mentioned below:
10. Being an MVC architecture framework, it ensures a tight separation between presentation layers and business logic. being based on MVC, the framework provides many features like high performance, increased security and scalability
11. This framework has integrated testing into it which makes sure that the application is bug free
12. It works on modular basis. It implies that with laravel, there are lot of pre-built functions and structures that work on the latest PHP principles to build responsive web applications in a time efficient manner.
13. It has better authentication and authorization option
14. It provides to build much more secure web application by protecting it against the most severe security risks: SQL injection, cross-site request forgery and cross-site scripting
15. It has automated and unit testing feature
16. It has automated task execution and scheduling feature.
17. It supports message queue system for load balancing. The system helps to maintain a healthy web server along with effective traffic management.
18. Application Type Choice: The built in application is web application which runs in any operating system and device so long they have a browser that supports the application.
19. Architecture Patterns: The application is built in MVC pattern which separates the representation of information from the user's interaction to it. it encourage segmentation of the code from different functional requirements, this means in essence three distinct parts dictate their uses.

**Model**: maintaining the application state

**Controller**: make changes to the application state

**View**: presents the application to the user

MVC ensures application flexibility, maintainability, ease of use and integration.

## 8.2. Network architecture capabilities

Below are the justification for the network architecture capabilities:

1. Bandwidth: The application will run on the
2. Network Failures : In case of the network failure, network administrator will provide the support to fix the problem
3. Loss of Data: Proper data backup mechanism will be implemented
4. Inconsistent data : The data will be inserted in the structured way
5. Inconsistent timing of data: The data access and output format will me properly defined
6. Cyber security: Proper security mechanism will be implemented

# 9. Technology Stack and Server Configuration

Below are the list of technology used :

1. Centralized E-Attendance Application
2. Attendance Device
3. Network Configuration with the Department Area
4. DOIT Server

Below are the details of server configuration :

1. We are deploying application to the server that is running Nginx. We use below configuration file as starting point of configuring web server.

server {

 listen 80;

 server\_name example.com;

 root /srv/example.com/public;

 add\_header X-Frame-Options "SAMEORIGIN";

 add\_header X-Content-Type-Options "nosniff";

 index index.php;

 charset utf-8;

 location / {

 try\_files $uri $uri/ /index.php?$query\_string;

 }

 location = /favicon.ico { access\_log off; log\_not\_found off; }

 location = /robots.txt { access\_log off; log\_not\_found off; }

 error\_page 404 /index.php;

 location ~ \.php$ {

 fastcgi\_pass unix:/var/run/php/php7.4-fpm.sock;

 fastcgi\_param SCRIPT\_FILENAME $realpath\_root$fastcgi\_script\_name;

 include fastcgi\_params;

 }

 location ~ /\.(?!well-known).\* {

 deny all;

 }

}

1. Autoloader Optimization: When deploying to production, we make sure that we are optimizing Composer's class autoloader map so Composer can quickly find the proper file to load for a given class:

 composer install --optimize-autoloader --no-dev

1. Configuration Loading Optimization: When deploying to production, we make sure that we run *confing:cache* Artisan command during deployment process.

php artisan config:cache

This command will combine all of Laravels configuration files into a single, cached file, which greatly reduces the number of trips the framework must make to the file system when loading configuration values.

1. Route Loading Optimization: When deploying, we make sure that we run *route:cache* Artisan command during deployment process.

php artisan route:cache

1. View Loading Optimization: When deploying, we make sure that we run *view:cache* Artisan command during deployment process.

php artisan view:cache

1. Debug Mode: The debug option in config/app.php configuration file determines how much information about an error is actually displayed to the user. by default, this option is set to respect value of the APP\_DEBUG environment variable, which is stored in .env file.

In the production environment, this value should always be false. If the APP\_DEBUG variable is set to true in production, we risk exposing sensitive configuration values to application's end users.

# 10. Standard and regulatory references

|  |  |  |
| --- | --- | --- |
| # | Document Identifier  | Document Title |
| [STD1] |  | Add your documents referencesOne line per document |

# 11. Annexure

## 11.1 Data Dictionary Template (Mandatory)

|  |  |
| --- | --- |
|  System: Centralized e-Attendance System | Database : doit |
| Overview |
| This database doit is RDBMS database designed for centralized e-attendance system. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Name | Description | Data Type | Constraint | Allow Null |
| Attendance Logs |  |
| id |  | int(11) | Primary Key | NO |
| device\_emp\_code |  | int(11) | Foreign Key | YES |
| ip\_address |  | varchar(200) | Foreign Key | YES |
| device\_id |  | int(11) | Foreign Key | YES |
| log\_time |  | datetime |  | YES |
| log\_types |  | tinyint(4) |  | YES |
| created\_at |  | datetime |  | YES |
| updated\_at |  | date |  | YES |
| Attendance  |  |
| id |  | int(11) | Primary Key | NO |
| emp\_id |  | int(11) | Foreign Key | YES |
| shift\_id |  | int(11) | Foreign Key | YES |
| device\_id |  | int(11) | Foreign Key | YES |
| attendance\_id |  | date |  | YES |
| in\_time |  | time |  | YES |
| out\_time |  | time |  | YES |
| create\_at |  | datetime |  | YES |
| update\_at |  | datetime |  | YES |
| Branch type |  |
| id |  | int(11) | PK | NO |
| org\_id |  | int(11) |  | YES |
| type\_name |  | varchar(255) |  | YES |
| type\_short\_name |  | varchar(50) |  | YES |
| active |  | Enum(‘Y’, ‘N’) |  | YES |
| Branches  |  |  |  |  |
| Id |  | int(11) | PK | NO |
| Org\_id |  | int(11) | FK | NO |
| Branch\_name |  | varchar(255) |  | NO |
| Branch\_name\_np |  | varchar(255) |  | NO |
| Created\_at |  | datetime |  | YES |
| Updated\_at |  | datetime |  | YES |
| countries |  |
| id |  | int(11) | PK | NO |
| country\_name |  | varchar(255) |  | YES |
| country\_name\_np |  | varchar(255) |  | YES |
| country\_code |  | varchar(20) |  | YES |
| country\_status |  | enum(‘Y’,’N’) |  | YES |
| created\_At |  | datetime |  | YES |
| updated\_at |  | datetime |  | YES |
| designations |  |
| id |  | int(11) | PK | NO |
| desg\_name |  | varchar(255) |  | YES |
| desg\_name\_np |  | varchar(255) |  | YES |
| desg\_order |  | nt(11) |  | YES |
| desg\_status |  | enum(‘Y’,’N’) |  | YES |
| created\_at |  | datetime |  | YES |
| updated\_at |  | datetime |  | YES |
| id |  | int(11) |  | NO |
| devices |  |
| id |  | int(11) | PK | NO |
| org\_id |  | int(11) | FK | YES |
| device\_name |  | varchar(255) |  | YES |
| model\_name |  | varchar(255) |  | YES |
| Ip\_address |  | varchar(20) |  | YES |
| Port\_number |  | varchar(4) |  | YES |
| Serial\_number |  | varchar(50) |  | YES |
| Device\_status |  | enum(‘Y’,’N’) |  | YES |
| Created\_at |  | datetime |  | YES |
| Deleted\_at |  | datetime |  | YES |
| Employee\_shifts |  |
| id |  | int(11) | PK | NO |
| emp\_id |  | int(11) | FK | NO |
| shift\_id |  | int(11) | FK | NO |
| start\_date |  | date |  | NO |
| end\_date |  | date |  | NO |
| create\_at |  | datetime |  | YES |
| update\_at |  | datetime |  | YES |
| Employee\_type |  |
| id |  | int(11) | PK | NO |
| Type\_name |  | varchar(255) |  | NO |
| Type\_name\_np |  | varchar(255) |  | NO |
| Type\_status |  | enum(‘Y’,’N’) |  | YES |
| Created\_at |  | datetime |  | YES |
| Updated\_at |  | datetime |  | YES |
| Employees  |  |
| id |  | int(11) | PK | NO |
| org\_id |  | Int(11) | FK | NO |
| org\_branch\_id |  | Int(11) | FK | NO |
| loignid |  | Varchar(255) |  | NO |
| User\_role\_id |  | Int(11) | FK | NO |
| User\_lang\_id |  | Int(11) | FK | NO |
| Device\_emp\_code |  | Int(11) |  | YES |
| Emp\_sanket\_number |  | Int(11) |  | YES |
| Desg\_id |  | Int(11) | FK | NO |
| Fullname\_eng |  | Varchar(255) |  | NO |
| fullname\_nep |  | Varchar(255) |  | NO |
| Gender |  | Enum(‘Y’,’N’) |  | NO |
| birth\_place |  | Varchar(255) |  | YES |
| mobile\_number |  | int(11) |  | YES |
| Email |  | Varchar(255) |  | YES |
| marital\_status |  | Enum(‘Y’, ‘N’) |  | NO |
| job\_appointment\_date |  | Date |  | YES |
| lang\_id |  | Int(11) | FK | YES |
| blood\_group |  | enum('A+','B+','AB+','O+','A-','B-','AB-','O-') |  | YES |
| religion\_id |  | Int(11) | FK | NO |
| Dob |  | Date |  | YES |
| Complexion |  | Varchar(255) |  | YES |
| is\_disabled |  | Enum(‘Y’, ‘N’) |  | NO |
| about\_employee |  | Int(11) |  | YES |
| Photo |  | Varchar(255) |  | YES |
| emp\_status |  | Enum(‘Y’, ‘N’) |  | YES |
| emp\_type\_id |  | Int(11) | FK | YES |
| created\_at |  | datetime |  | YES |
| updated\_at |  | datetime |  | YES |
| Kaaj\_applications |  |
| Id |  | Int(11) | PK | NO |
| Emp\_id |  | Int(11) | FK | NO |
| Emp\_org\_id |  | Int(11) | FK | YES |
| Emp\_branch\_id |  | Int(11) | FK | YES |
| Emp\_type\_id |  | Int(11) | FK  | YES |
| Emp\_desg\_id |  | Int(11) | FK | YES |
| Kaaj\_type\_id |  | Int(11) | FK | YES |
| Year\_id |  | Int(11) | FK | YES |
| Kaaj\_title |  | text |  | YES |
| Kaaj\_details |  | text |  | YES |
| Kaaj\_paid |  | enum('Y','N') |  | YES |
| Is\_national |  | enum('Y','N') |  | YES |
| Start\_date |  | date |  | YES |
| End\_date |  | Date |  | YES |
| Vehicle |  | Varchar(255) |  | YES |
| Kaaj\_country\_id |  | int(11) | FK | YES |
| Objective |  | Text |  | YES |
| Venue |  | Text |  | YES |
| Kaaj\_file |  | Varchar(255) |  | YES |
| Kaaj\_status |  | Enum(‘Draft’, ‘Applied’, ‘Canceled’) |  | YES |
| Approver\_id |  | Int(11) | FK | YES |
| Approver\_status |  | Enum(‘Applied’, ‘Approved’, ‘Rejected’, ‘Canceled’) |  | YES |
| Approver\_status\_date |  | Datetime |  | YES |
| Applied\_date |  | Date |  | YES |
| Created\_at |  | Datetime |  | YES |
| Updated\_at |  | Datetime |  | YES |
| Kaaj types |  |
| Id |  | Int(11) | PK | NO |
| Type\_name |  | Varchar(255) |  | YES |
| status |  | Enum(‘Y’, ‘N’) |  | YES |
| Created\_at |  | Datetime |  | YES |
| Updated\_at |  | Datetime |  | YES |
| languages |  |
| Id |  | Int(11) | PK | NO |
| Lang\_name |  | Varchar(255) |  | YES |
| Lang\_status |  | Enum(‘Y’, ‘N’) |  | YES |
| Created\_At |  | Datetime |  | YES |
| Updated\_at |  | Datetime |  | YES |
| Leave applications  |  |
| id |  | int(11) | PK | NO |
| emp\_id |  | int(11) | FK | NO |
| emp\_type\_id |  | int(11) | FK | NO |
| desg\_id |  | int(11) | FK | NO |
| org\_id |  | int(11) | FK | NO |
| org\_section\_id |  | int(11) |   | NO |
| year\_id |  | int(11) | FK | NO |
| leave\_apply\_date |  | date |   | NO |
| leave\_type\_id |  | int(11) | FK | NO |
| leave\_half\_day |  | enum('Y','N') |   | NO |
| leave\_start\_date |  | date |   | YES |
| leave\_end\_date |  | date |   | YES |
| leave\_status |  | enum('Draft','Applied','Canceled') |   | NO |
| leave\_file |  | varchar(255) |   | YES |
| leave\_reason |  | text |   | YES |
| recommender\_id |  | int(11) | FK | NO |
| recommender\_status |  | enum('Applied','Approved','Rejected','Canceled') |   | YES |
| recommender\_status\_date |  | date |   | YES |
| recommender\_details |  | text |   | YES |
| approver\_id |  | int(11) | FK | YES |
| approver\_status |  | enum('Approved','Rejected') |   | YES |
| approver\_status\_date |  | date |   | YES |
| approver\_details |  | text |   | YES |
| created\_at |  | date |   | YES |
| updated\_at |  | date |   | YES |
| Leave balance |  |
| Id |  | Int(11) | PK | NO |
| Emp\_id |  | Int(11) | FK | NO |
| Year\_id |  | Int(11) | FK | NO |
| Leave\_type\_id |  | Int(11) | Fk | NO |
| Days\_number |  | Decimal(8,2) |  | NO |
| Leave titles  |  |
| id |  | Int(11) | PK | NO |
| title |  | varchar(255) |  | YES |
| title\_short |  | varchar(50) |  | YES |
| title\_np |  | varchar(255) |  | YES |
| title\_short\_np |  | varchar(50) |  | YES |
| status |  | enum('Y','N') |  | YES |
| Leave types  |  |
| id |  | int(11) | PK | NO |
| ord\_id |  | int(11) |  | YES |
| leave\_title\_id |  | int(11) |  | YES |
| male\_only |  | enum('Y','N') |  | YES |
| fremale\_only |  | enum('Y','N') |  | YES |
| contract\_eligible |  | enum('Y','N') |  | YES |
| weekend\_countable |  | enum('Y','N') |  | YES |
| public\_holiday\_countable |  | enum('Y','N') |  | YES |
| saving\_leave |  | enum('Y','N') |  | YES |
| recurrance\_leave |  | enum('Y','N') |  | YES |
| office\_head\_approval |  | enum('Y','N') |  | YES |
| single\_leave |  | enum('Y','N') |  | YES |
| deducate\_from\_other\_leave |  | enum('Y','N') |  | YES |
| half\_day\_apply |  | enum('Y','N') |  | YES |
| recommedner\_required |  | enum('Y','N') |  | YES |
| satta\_bida |  | enum('Y','N') |  | YES |
| active |  | enum('Y','N') |  | YES |
| max\_saving |  | int(11) |  | YES |
| total\_days |  | int(11) |  | YES |
| number\_of\_times |  | int(11) |  | YES |
| extra\_days |  | int(11) |  | YES |
| apply\_docment\_within\_days |  | int(11) |  | YES |
| approve\_within\_days |  | int(11) |  | YES |
| created\_at |  | datetime |  | YES |
| updated\_at |  | datetime |  | YES |
| Leaves  |  |
| id |  | int(11) | PK | NO |
| org\_id |  | int(11) |  | YES |
| leavel\_name |  | varchar(255) |  | YES |
| level\_short\_name |  | varchar(255) |  | YES |
| display\_order |  | enum('Y','N') |  | YES |
| higer\_post |  | enum('Y','N') |  | YES |
| middle\_post |  | enum('Y','N') |  | YES |
| lower\_post |  | enum('Y','N') |  | YES |
| active |  | enum('Y','N') |  | YES |
| Organization types  |  |
| id |  | int(11) | PK | NO |
| type\_name |  | varchar(255) |  | YES |
| type\_name\_np |  | varchar(255) |  | YES |
| type\_status |  | enum('Y','N') |  | YES |
| created\_at |  | datetime |  | YES |
| updated\_at |  | datetime |  | YES |
| Organizations  |  |
| id |  | int(11) | PK | NO |
| org\_type\_id |  | int(11) | FK | NO |
| org\_name |  | varchar(255) |  | YES |
| org\_name\_np |  | varchar(255) |  | YES |
| org\_code |  | varchar(20) |  | YES |
| org\_code\_np |  | varchar(20) |  | YES |
| main\_org\_id |  | int(11) |  | YES |
| org\_fax |  | varchar(20) |  | YES |
| org\_phone |  | varchar(50) |  | YES |
| org\_pradesh\_id |  | int(11) |  | YES |
| org\_address |  | varchar(255) |  | YES |
| org\_address\_np |  | text |  | YES |
| org\_remarks |  | text |  | YES |
| org\_main\_office |  | enum('Y','N') |  | YES |
| org\_status |  | enum('Y','N') |  | YES |
| Pradesh  |  |
| id |  | int(11) | PK | NO |
| pradesh\_name |  | varchar(255) |  | YES |
| pradesh\_name\_np |  | varchar(255) |  | YES |
| created\_at |  | datetime |  | YES |
| updated\_at |  | datetime |  | YES |
| Public holidays  |  |
| id |  | int(11) | PK | NO |
| org\_id |  | int(11) |  | YES |
| holiday\_name |  | varchar(255) |  | YES |
| holiday\_name\_short |  | varchar(50) |  | YES |
| start\_date |  | date |  | YES |
| end\_date |  | date |  | YES |
| for\_male |  | enum('Y','N') |  | YES |
| for\_female |  | enum('Y','N') |  | YES |
| for\_only\_disable |  | enum('Y','N') |  | YES |
| religion\_id |  | int(11) |  | YES |
| active |  | int(11) |  | YES |
| Religions  |  |
| id |  | int(11) | PK | NO |
| religion\_name |  | varchar(255) |  | YES |
| religion\_status |  | enum('Y','N') |  | YES |
| created\_at |  | datetime |  | YES |
| updated\_at |  | datetime |  | YES |
| Roles  |  |
| id |  | int(11) | PK | NO |
| role\_name |  | varchar(255) |  | NO |
| role\_name\_np |  | varchar(255) |  | YES |
| role\_status |  | enum('Y','N') |  | NO |
| created\_at |  | datetime |  | YES |
| Service types  |  |
| id |  | int(11) | PK | NO |
| org\_id |  | int(11) |  | YES |
| service\_types |  | varchar(255) |  | YES |
| service\_types\_short |  | varchar(255) |  | YES |
| display\_order |  | int(11) |  | YES |
| permanent\_available |  | enum('Y','N') |  | YES |
| active |  | enum('Y','N') |  | YES |
| Shift days  |  |
| id |  | int(11) | PK | NO |
| shift\_id |  | int(11) | FK | YES |
| day\_name |  | varchar(3) |  | YES |
| start\_time |  | time |  | YES |
| end\_time |  | time |  | YES |
| is\_holiday |  | enum('Y','N') |  | YES |
| created\_at |  | datetime |  | YES |
| updated\_at |  | datetime |  | YES |
| Shifts  |  |
| id |  | int(11) | PK | NO |
| org\_id |  | int(11) | FK | YES |
| shift\_name |  | varchar(255) |  | YES |
| shift\_status |  | enum('Y','N') |  | YES |
| created\_at |  | datetime |  | YES |
| Years  |  |
| id |  | int(11) | PK | NO |
| year\_name |  | varchar(50) |  | NO |
| year\_name\_np |  | varchar(50) |  | NO |
| eng\_start\_date |  | date |  | NO |
| eng\_nep\_date |  | datetime |  | NO |
| running\_year |  | enum('Y','N') |  | NO |
| year\_status |  | enum('Y','N') |  | YES |
| created\_at |  | datetime |  | YES |